REMARKS

Claims 1-16 are currently pending in the patent application. The Examiner has finally rejected Claims 1-16 under 35 USC 102 as anticipated by Faiman. The Examiner has noted that the claims did not directly reflect the language of the Applicants' arguments. Applicants have amended the language of many of the pending claims to expressly recite the dynamic compiler and the dynamic compiling steps. Based on those amendments, and for the reasons set forth below, Applicants believe that all of the pending claims are patentable over the Faiman reference.

The present invention is directed to a dynamic compiler, a dynamic compiling method, a computer, a storage medium and a support program for optimizing a program during compiling thereof. During translation from source code to machine code, the present invention performs a dynamic analysis during execution to determine whether the execution speed of the program can be increased by fixing, in a specific state, a parameter for a predetermined command in the program; and then employs the results of the analysis to generate a path along which the parameter of the predetermined command is fixed in the specific state. Since the present inventive approach generates the path along

which the parameter will be fixed in the specific state, the present invention provides a dynamic compiler method which provides for dynamic specialization in the instances when the call method cannot be specified at a location whereat the method call is to be issued.

As is detailed in the Background section of the present Specification, the method which is taught and claimed improves on the prior art specialization methods which are done by static compiler (e.g., page 1, lines 19-23). Under one prior art method, a static compiler detects an argument that always has the same constant value and then substitutes that constant value for the argument at compile time. Under the other prior art method, a static compiler performs the same substitution for each nested loop at compile time and a specialization method is performed at the location where a method call should be issued. Neither of the foregoing prior art methods are appropriate for a dynamic compiler or for a program having a dynamic call function such that different values can be dynamically designated for specialization, or path generation, based on a dynamic analysis of execution performance.

The pending claims expressly provide that the method comprises performing a dynamic analysis to determine whether the execution speed of the program can be increased by

fixing, in a specific state, a parameter for a predetermined command in the program, and then employing the results of the analysis for the generation, in said program, of a path along which said parameter of the predetermined command is fixed in that specific state. Clearly the language is not claiming a static substitution of a predetermined value at compile time but is claiming a process for analyzing execution performance/statistics and generating a path for fixing one of the parameters dynamically during program compilation.

The cited Faiman patent is directed to constructing more efficient compilers for compiling code which can run on many different machines. Faiman effectively separates the compiler functions into front end functions and back end functions, wherein the front end directly accesses the source code module and is very language-specific whereas the back end functions are machine-specific. Faiman provides that "[t]he front end translates programs from the source language into the intermediate language...and the back end translates programs from the intermediate language into the target machine language" (see: Col. 21, lines 63-67). In order to compile the code, the Faiman approach is to translate the source code into an intermediate code which is graphed. The intermediate code graph can then serve as an

interface between the generic back end and the language-specific front end of the compiler. Faiman teaches that optimization is one of the back end operations. As detailed in Col. 22, lines 7-20, the compiler performs a constant expression evaluation routine to detect constants, calculate them at compile time, and "fold" them into the object code image (see: Col. 22, lines 17-19). This is the prior art optimization by a static compiler which is referred to above and in the present Specification. Faiman acknowledges that the implementation of a constant expression evaluation routine may be "a matter of considerable difficulty" (Col. 22, lines 47-48) due to the fact that Faiman is providing this intermediate language graph as the input to the back end compiler functions. Accordingly, Faiman provides an additional translation approach, detailed in Col. 23, lines 11-35, wherein a special front end compiler generates the intermediate language for the constant expression evaluation routine, creates the intermediate graph with the translated operators for performing the routine, and then generates machine code for the constant expression evaluation routine. Once the machine code version of the constant expression evaluation routine has been generated, the aforementioned static compiler approach to program optimization can be performed.

Applicants respectfully assert that the Faiman patent does not teach or suggest the invention as claimed. While Faiman does discuss optimization using constant expression evaluation, it is the prior art static optimization approach that is used in Faiman. Faiman does not perform any dynamic analysis during execution to determine whether the execution speed of a program can be increased by fixing, in a specific state, a parameter for a predetermined command in said program. Rather, Faiman operates on the assumption that eliminating any fetch operations will automatically result in reduced speed (see: Col. 22, lines 12-16). Faiman does not teach or suggest that parameters be fixed for specific states, since Faiman assumes a static compiler. Further, Faiman does not teach or suggest generation of a path along which a parameter of a predetermined command is fixed in a specific state. Faiman statically substitutes a constant for an expression, regardless of relative costs/speed implications, regardless of state, and regardless of path. Accordingly, Applicants conclude that Faiman does not anticipate the language of Claims 1-2, 5-10, 13 and 15.

Applicants further assert that the Faiman teachings regarding analyzing induction variables (Col. 3, line 65-Col. 4, line 5 and Col. 18, lines 40-67) do not anticipate or render obvious the claimed steps of employing

analysis results to generate a path along which a parameter is fixed in a specific state, let alone of obtaining statistical data for the appearance frequency of each available state for the program parameter. What Faiman is teaching is the substitution of addition operations in place of multiplication operations. Moreover, Faiman expressly states that a thorough analysis is too costly (see: Col. 4, lines 10-14 and Col. 20, lines 60-62) and that a simple side effects approach is used, predicated on an assumption that "all variables modified...are basic induction variables" (Col. 21, lines 12-13). Clearly such teachings do not anticipate an express recitation of obtaining statistical data for the appearance frequency of each available state (Claims 2-4, 8-9, 11-12, 14 and 16).

Applicants respectfully assert that the Faiman patent does not teach or suggest the invention as claimed. It is well established under U. S. Patent Law that, for a reference to anticipate claim language under 35 USC 102, that reference must teach each and every claim feature. Since the Faiman patent does not teach steps or means for translating and optimizing a program comprising steps of performing a dynamic analysis during execution to determine whether the execution speed of said program can be increased by fixing, in a specific state, a parameter for a

predetermined command in said program; employing results of an analysis for the generation, in said program being compiled, of a path along which said parameter of said predetermined command is fixed in said specific state, or obtaining statistical data for appearance frequency for each available state and using that statistical data for dynamically generating a path along which the parameter of a particular command is fixed in that state, it cannot be maintained that Faiman anticipates the invention as set forth in the independent claims, Claims 1, 3, 5, 6, 7, 10, 11, and 13-16, or the claims which depend therefrom and add further limitations thereto.

Based on the foregoing amendments and remarks, Applicants respectfully request entry of the amendments, reconsideration of the amended claim language in light of the remarks, withdrawal of the rejections, and allowance of the claims.

Respectfully submitted,

M. Kawahito, et al

By: Anne Vachon Dougherty

Anne Vachon Dougherty
Registration No. 30,374
Tel. (914) 962-5910